Maastricht University

BPMN: A meta model for the Happy Path

Citation for published version (APA):

Bollen, P. W. L. (2010). *BPMN: A meta model for the Happy Path*. METEOR, Maastricht University School of Business and Economics. METEOR Research Memorandum No. 003
https://doi.org/10.26481/umamet.2010003

Document status and date:
Published: 01/01/2010

DOI:
10.26481/umamet.2010003

Document Version:
Publisher's PDF, also known as Version of record

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.umlib.nl/taverne-license

Take down policy
If you believe that this document breaches copyright please contact us at:

repository@maastrichtuniversity.nl

providing details and we will investigate your claim.

Download date: 05 May. 2023

Maastricht University

Peter Bollen

BPMN: A Meta Model for the Happy Path

RM/10/003

METEOR

Maastricht University School of Business and Economics
Maastricht Research School of Economics
of Technology and Organization

P.O. Box 616
NL - 6200 MD Maastricht
The Netherlands

BPMN: A Meta Model for the Happy Path

Peter Bollen
Department of Organization & Strategy
School of Business and Economics
Maastricht University,
The Netherlands

Abstract. Recently, the OMG has been working on developing a new standard for a business process management notation (BPMN). This standard development results in documents that contain the newest approved version of a standard or a standard proposal that can be ammended. It is our vision that such a standard document, that also serves as a specification for BPMN modeling tool developers could benefit from a fact-oriented model in which the same domain knowledge is represented conceptually as a list of concept definitions (including naming conventions), a set of information structure diagrams and the constraints or business rules that govern the instances of the information structure diagrams. In this paper we will show precisely, how such a fact-oriented conceptual view on a standard document can be created, and we will show how a fact-oriented approach can improve the completeness of a specification.

1. Introduction

In the 1970's and 1980's a long-standing discussion took place on the stance that IS developers should have towards approaching IS specifications from a scientific point of view: *data-oriented* or *process-oriented*. Over the past decade or so the discussions have changed because of the emergence of UML in which technically, the data-oriented and process-oriented aspects are adressed 'equally'. At the same time, though, the field of IS specification has undergone a dramatically shift towards 'user-friendly' and 'business-language oriented' modeling languages like SBVR [1] and BPMN [2, 3].

In recent years OMG has been working on a standard for a Business Process Management Notation (BPMN), e.g. see [2]. Although the development and standardization of a new business process modeling language of such a major standardization organization as OMG is welcomed by the scholars and practitioners of business process modeling we think that the achievements in the field of fact-oriented conceptual modeling should be applied to the BPMN standard documents as well. It is our vision that such a standard document, that also serves as a specification for BPMN modeling tool developers could benefit from a fact-oriented model in which the domain knowledge is represented conceptually as a list of concept definitions (including naming conventions), a set of information structure diagrams and the constraints or business rules that govern the instances of the information structure diagram.

In this paper we will analyze the BPMN 1.1 standard in combination with rules and guidelines on how to model appropriate BPMN models as is illustrated in chapters 1 through 6 of the book by Bruce Silver [4].

In section 2 we will give an introduction to the BPMN 1.1 modeling constructs as given in the standard document [2]. In section 3 we will give the BPMN meta-model for Silver's level 1 palette of constructs [4]. In the meta model we will also add the modeling guidelines and constraints that are explained in Silver's level 1 BPMN method and BPMN style [4]. We will use the fact-oriented conceptual modeling methodology [5, 6] to express the BPMN meta model.

1.1 Introduction to the Fact-Oriented Modeling Methodology

Fact-Oriented Modeling (FOM) is a methodology for modeling domain knowledge on the conceptual level. It is named after its main concepts: facts and fact types. FOM is applicable for all verbalizable knowledge sources. A verbalizable knowledge source is a document that often is incomplete, informal, ambiguous, possibly redundant and possibly inconsistent. As a result of applying the fact-oriented knowledge extracting procedure (KEP) [7, 8] we can create a document that only contains structured

knowledge or a knowledge grammar which structures verbalizable knowledge into the following elements (*knowledge reference model(KRM)*):

1. Knowledge domain sentences
2. Definitions and naming conventions for concepts used in domain sentences
3. Knowledge domain fact types including sentence group templates
4. Population state (transition) constraints for the knowledge domain
5. Derivation rules that specify *how* specific domain sentences can be derived from other domain sentences.
6. Rules that specify *what* fact instances can be inserted, updated or deleted.
7. Event rules that specify *when* a fact is derived from other facts or when a fact must be inserted , updated or deleted.

A KRM of a subject domain potentially can contain hundreds, possibly thousands of concept definitions, naming conventions, fact types, population constraints, derivation rules and event rules. The knowledge extracting procedure (KEP) specifies *how* we can transform an informal, mostly incomplete, mostly undetermined, possibly redundant and possibly inconsistent description of domain knowledge into the following classes: *informal comment*, *non-verbalizable knowledge* and *verbalizable knowledge* to be classified into types 1 through 7 of the KRM. We note that the sub-procedure that is needed to instantiate the elements 1 through 5 (of the KRM) is known as conceptual schema design procedure (CSDP) [9].

In the fact-oriented approach, the fact construct is used for encoding all semantic connections between entities. The 'role-based' notation makes it easy to define static constraints on the data structure and it enables the modeler to populate conceptual schemas with example sentence instances for constraint validation purposes. The fact-oriented modeling approach has its roots in the seventies and over the years a number of dialects have evolved, i.e. ENALIM [10], (binary) NIAM [11], N-ary NIAM [6], Fully Communication Oriented Information Modeling (FCO-IM) [12], Object-Role Modeling[9] and CogNiam [13]. The OMG business rule standard SBVR can be considered the latest fact-oriented dialect specialized for declaring business rules [1]. The fact-oriented 'dialect' that we will use in this article is a combination of CogNIAM [14] and SBVR [15] for the list of concept definitions and naming conventions and the expression of concepts, ground facts, fact types and business rules in structured natural language [16]. As a graphical notation for the fact types and business rules we will use Object-Role Modeling's notational conventions [9].

Figure 1 summarizes the notational conventions in the Fact-Oriented ORM modeling dialect that we will use in this paper.

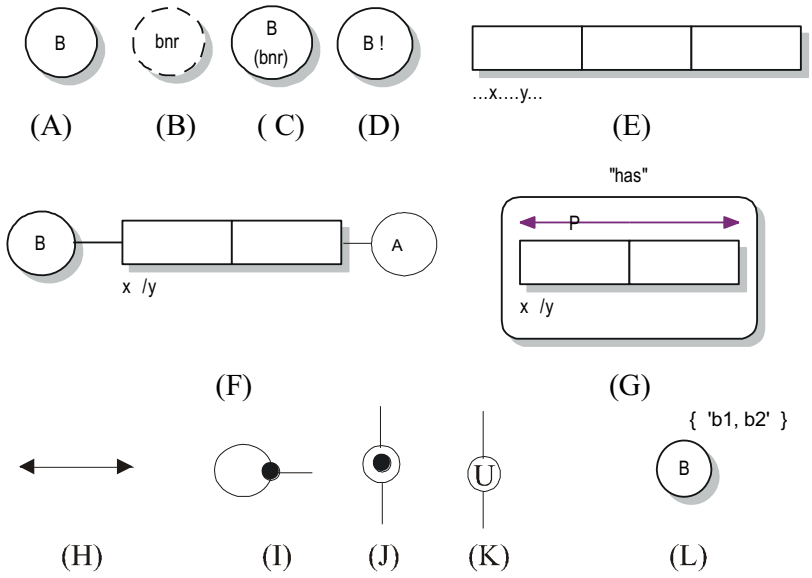

Fig. 1. Main notation symbols in the ORM fact-Oriented dialect

Atomic *entities* (figure 1A) or *data values* (figure 1B) are modeled in ORM as simple (hyphenated) circles. Instances of an entity type furthermore can exist independently (e.g. they are not enforced to participate in any relationship), which is shown by adding an exclamation point after the entity type's name (figure 1D). *Simple* reference schemes in ORM are abbreviated by putting the *value type* or *label type* in parenthesis beneath the name of the entity type (figure 1C). Semantic connections between entities are depicted as combinations of boxes (figure 1E) and are called *facts* or *fact types* in ORM. Each box represents a role and must be connected to either an *entity type*, a *value type* or *a nested object type* (see figure 1F). A fact type can consist of one or more roles. The number of roles in a fact type is called the fact type arity. The semantics of the fact type are put in the *fact predicate* (this is the text string *...x...y...* in figure 1E). A *nested object type* (see figure 1G) is a non-atomic entity type that is connected to a fact type that specifies what the constituting entity types and/or values types are for the nested object type.

Figures 1H through 1L illustrate the diagramming conventions for a number of *static population constraint(s) (types)* in ORM. A double-arrowed line (figure 1H) that covers one or more 'boxes' of a fact type is the symbol for an *internal uniqueness constraint*. The symbol in figure 1K stands for an *external uniqueness constraint*. A(n) uniqueness constraint restricts the number of identical instances of a role combination 'under' the uniqueness constraint to *one*. A *mandatory role constraint* (figure 1I) can be added to a role. It specifies that each possible instance of such an object type must play that designated role at *all* times. A *disjunctive mandatory role constraint* (figure 1J) is defined on two or more roles and specifies that each possible instance of the object type connected to these roles must *at least* play *one* of these roles at *any* time. In figure 1L an example of a value constraint is given that enforces that each instance of the object type B either has the value *b1* or *b2*.

2. The modeling of Silver's level 1 palette

In order to precisely show how a business data –or information model is needed to create well-formed and well-integrated BPMN models we will start with the introduction of the BPMN modeling constructs that enable us to model the 'happy path', e.g. those sequences of activities that will be executed if everything goes as expected without exceptions [4]. We will restrict BPMN at this point to those modeling constructs

that comprise Silver's level 1 palette [4]: Pool and Lane, User and Service Task, (collapsed and expanded) Subprocess, Start Event, End Event, Exclusive and Parallel Gateway, Sequence Flow and Message Flow, Data Object and Message Flow.

2.1 Pools and Lanes

In BPMN two types of *swimlane* constructs can be used: *pools* and *lanes*. The *pool* concept allows modelers to specify message flows between two (or more) separate business entities. The *lane* construct can be used to show how certain activities are associated with a specific company function [3, p. 4-5]. A distinction is made into *white box* and *black box* pools [4]

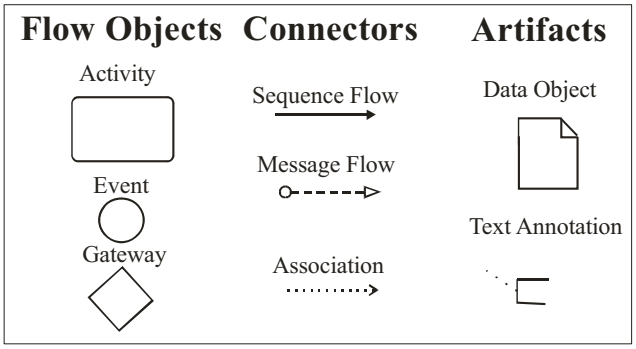

Fig. 2. Diagrammatic representation of most important BPMN modeling concepts

'An *event* is something that happens during the course of a business process' [2, p.18]. 'An *activity* is a generic term for work that a company performs…The types of activities that are a part of a process model are: *process, sub-process* and *task*. [2, p.18]. 'A *gateway* is used to control the divergence and convergence of sequence flow' [2, p.18].'A sequence flow is used to show the order that activities will be performed in a process' [2, p.19]. 'An *association* is used to associate information flow with flow objects' [2, p.19]. '*data objects* are considered artifacts because they do not have any direct effect on the sequence flow or message flow of the process, but they do provide information about what activities required to be performed and/or what they produce' [2, p.19]. In figure 2 we have depicted the graphical representations of the most important BPMN modeling constructs.

2.2 Activities: Subprocesses and Tasks

An activity is work that is performed within a business process. A *task* is an atomic activity that can not be broken down to (a) finer level of activity [3]. A *sub-process* is a non-atomic or compound activity that can be broken down into a set of sub-activities [3].

2.3 Start - and End-Events

For a number of sub-types of BPMN modeling concepts additional attributes are defined. For example for the *event* modeling concept we have as an attribute the event type, that must either have the value: *start, end* or *intermediate.*

2.4 Gateways, Sequence Flow and Message Flow

Gateways in BPMN are used to control the sequence flow in terms of convergence and divergence and a gateway has a number of attributes. There are basically three types of connectors in BPMN: *sequence flow*, *message flow* and an *association.* A *sequence flow* depicts the order in which the connected activities are performed. A *message flow* shows a flow of messages between two objects. An association is used to relate *artifacts* with *flow objects* and is mainly used to show the (data) *inputs* and (data) *outputs* of *activities*.

2.5 Data Object, Data Store, and Message

A *data object* is one of the three artifact types that are currently defined in the BPMN standard [2]. The 'incorporation' or 'leaving-out' of data-objects in a process model, is a way for some modelers to leave out 'clutter' [2, p.93].

3. A fact-based meta model for the palette 1 modeling constructs in BPMN.

In section 2 we have provided the definitions of the basic BPMN modeling concepts. In this section we will give an overview of the UoD that consists of the allowed BPMN expressions based upon the example that Silver uses to describe 'level-1' BPMN models [4]. This will be the starting point, for the derivation of a fact-oriented BPMN meta model by giving 'positive examples' that will lead us to the object types and fact types in the meta model. Furthermore, the modeling rules and constraints and the 'non-allowed' examples in the defining and practitioner's literature [2, 4] will lead to the definition of population constraints in the fact-oriented BPMN meta model. As a starting point for the fact-oriented analysis of BPMN example models we will use figure 5.7 on page 46 of Silver [4].

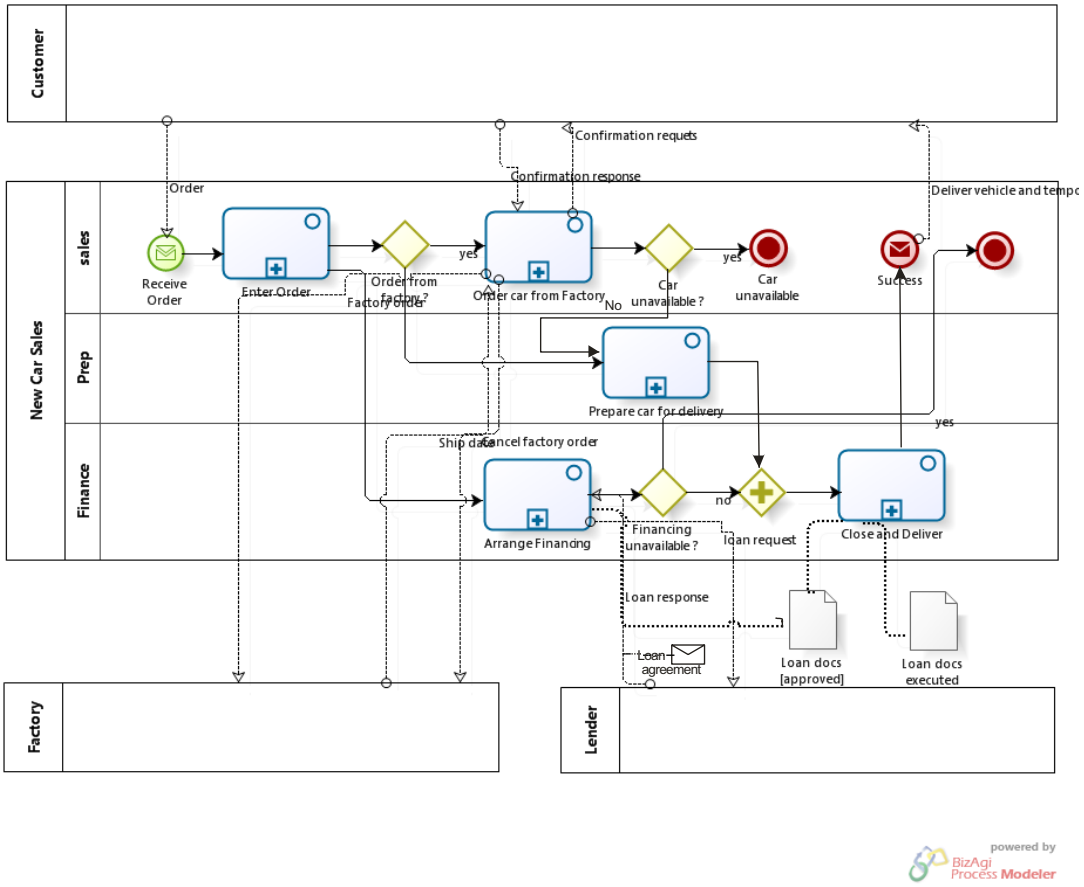

Fig 3. Example BPMN model as in Silver [4]

3.1 The list of concept definitions and naming conventions

In this section we will group and synthesize the definitions of the main modeling constructs in the BPMN in line with the definitions as it can be found in the OMG standard document [2] and the modeling guidelines and naming conventions as they are recommended in Silver [4]. We create this list of definitions by scanning the standard and descriptive documents of BPMN for definitions and explanations.

We have taken these definitions and we have incorporated them, together with other definitions into our list of concept definitions in table 1.

Table 1.: List of definitions for the BPMN standard, method and style in order of comprehension

Concept	Definition
Activity	An activity is work that is performed within a business
Process	A Process is any [Activity] performed within or across companies or organization (OMG v 1.1., p.32)
Process name	A name that designates a specific [Process] among

	the union of [Process]es
SubProcess	A SubProcess is a [Process] that is included within another [Process] (OMG v1.1, p. 287)
Task	A Task is an atomic [Activity] (Silver, 2009, p.27)
Task name	A name that designates a specific [Task] among the union of [task]s within a [SubProcess]. The names should, preferably have the following format; VERB-NOUN (Silver, 2009, p.27)
User Task	A User Task is a [Task] performed by a person, ie. A human activity (Silver, 2009, p.27)
Service Task	A Service Task is a [Task] in the form of an automated activity
Event	An Event is something that "happens" during the course of a business process (OMG v1.1, p. 18)
Event name	A name that designates a specific [Event] among the union of [Events]s within a given [Lane] of a given [Pool].
Start Event	A Start Event indicates the start of a [Process] or [SubProcess] (Silver, 2009, p.28)
None Start Event	A None Start Event is an [Event] of which the trigger is not specified. (Silver, 2009, p.28)
Message Start Event	A Message Start Event is an [Event] that is triggered by a signal outside the [Process]
Timer Start Event	A Timer Start Event signifies the [Process] is run on some predetermined schedule, either one-time or recurring (Silver, 2009, p.28)
End Event	An End Event indicates the end of a path in a [Process] or a [SubProcess] (Silver, 2009, p.28)
None End Event	A None End Event signifies that no result signal is thrown when the [End Event] is reached (Silver, 2009, p.28)
Message End Event	A Message End Event signifies that a message is sent when the [End Event] is reached (Silver, 2009, p.28)
Terminate End Event	A Terminate End Event immediately ends the [Process] or [SubProcess] in which it is contained. (Silver, 2009, p.29)
Pool	A Pool represents a participant in a [Process] (OMG, v1.1, p.87)
Pool name	A name that designates a specific [Pool] among the union of [Pool]s.
Lane	A Lane is a sub-partition within a [Pool] (OMG, v1.1, p.89)
Lane name	A name that designates a specific [Lane] among the union of [Lane]s within a [Pool].
White-Box Pool	A White-Box Pool is a [Pool] that contains a single [Process] (Silver, 2009, p.26)
White-Box Pool name	A [Pool name] that designates a specific [Process] among the union of [Process]es
Sequence Flow	A Sequence Flow shows the order in which the

	connected [Activities] will be performed in a [Process] (OMG v.1.1, p. 287)
Black-Box Pool	A Black-Box Pool is a [Pool] that represents a participant in the Collaboration and does not have an associated [Sequence Flow] (Silver, 2009, p.28; OMG, v1.1, p.87)
Gateway	A Gateway controls how the [Sequence Flow] interact as they converge and diverge within a [Process] (OMG v.1.1, p.70)
Gateway descriptor	A name that designates a specific [Gateway] among the union of [Gateway]s within a [Lane].
Exclusive Gateway	An Exclusive gateway is a [Gateway] that represents an exclusive decision. This means that only one of the [Sequence Flow]s is to be followed (Silver, 2009, p.29) .
Parallel Gateway	A Parallel Gateway is a [Gateway] that represents that all of the outgoing [Sequence Flow]s are to be followed in parallel. (Silver, 2009, p.29) .
Message	A message is a [Data Association] to a [Message Flow] (Silver, 2009, p.32)
Message Flow	A Message Flow represents a [Message] or a Signal sent between two [Pool]s. (Silver, 2009, p.30)
Message Flow name	A name that designates the content of a specific [MessageFlow] among the union of contents of[MessageFlow]s within a [Process].
Data Object	A variable inside the [Process] (Silver, 2009, p.31)
Data Association	Connects a [DataObject] to an [activity]
Non-Directional data association	Connects a [DataObject] to an [activity]
Directional data association	Connects an outgoing [DataObject] to an [activity] OR Connects an ingoing [DataObject] to an activity

3.2 Identification and naming conventions

In this paper we will use 'local' identifiers for domain concepts. This means that we have to give specific naming rules for domain concepts. These domain rules should be in line with the definition of the name classes in the list of concept definitions. E.g., the name class *event name* is defined as follows (see also table 1) :

'A name that designates a specific [Event] among the union of [Events]s within a given [Lane] of a given [Pool].'

This means that in verbalizing instances of a(n) (type of) event in a BPMN diagram, we should use a compound identifier as follows:

' The Terminate End Event 'Success' within the Lane 'Sales' of the White-Box Pool 'New Car Sales'

Another assumption, for naming conventions, that could have been chosen is to use 'global' unique identifiers for instances of concepts/object types. In most cases, however, this means that we need to

introduce some form of abstract object ID that bears no single resemblance to the practice of naming these concepts in the subject domain itself. We note that the definition of our naming conventions includes the case in which 'global' identifiers are used ("global uniqueness implies local uniqueness").

3.3 Verbalization of examples into elementary ground facts

Now the list of concept definitions is complete, we will take the example that is depicted in figure 5.7 of [4]. Applying step 1 ('verbalization' or 'from examples to elementary facts') from the fact-oriented conceptual schema design procedure (while at the same time using the defined concepts and naming conventions from the list of concept definitions) [5, 6, 17] will lead to the following fact verbalizations as SBVR ground facts:

The **Black-Box Pool** 'Customer' has an outgoing **MessageFlow** 'Order' to the **Message Start Event** 'Receive Order' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales'.

The **Black-Box Pool** 'Customer' has an outgoing **MessageFlow** 'confirmation response' to the **SubProcess** 'Order Car from Factory'.

The **SubProcess** 'order Car from Factory' has an outgoing **MessageFlow** 'confirmation request' to the **Black-Box Pool** 'Customer'

The **Message End Event** 'Success' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing **MessageFlow** 'Deliver vehicle and temporary registration' to the **Black-Box Pool** 'Customer'

The **Black-Box Pool** 'Lender' has an outgoing **MessageFlow** 'Loan response' to the **SubProcess** 'Arrange Financing' .

The **Message Flow** 'loan response' has the **Message** 'loan agreement' connected to it.

The **Black-Box Pool** 'Factory' has an outgoing **MessageFlow** 'Ship date' to the **SubProcess** 'Order Car From Factory'.

The **SubProcess** 'Order Car from factory' has an outgoing **MessageFlow** 'Factory order' to the **Black-Box Pool** 'Factory'.

The **SubProcess** 'order Car from Factory' has an outgoing **MessageFlow** 'Cancel Factory order' to the **Black-Box Pool** 'Factory'.

The **SubProcess** 'Arrange Financing' has an outgoing **MessageFlow** 'Loan request' to the **Black-Box Pool** 'Lender'.

the **Message Start Event** 'Receive Order' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing Sequence Flow to the **SubProcess** 'Enter Order'.

The **SubProcess** 'Arrange Financing' is located within the **Lane** 'Finance' of the **White-Box Pool** 'New Car Sales'

The **SubProcess** 'Enter Order' is located within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales'

The **SubProcess** ‘Order Car from Factory’ is located within the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car Sales’

The **SubProcess** ‘Pepare Car for Delivery’ is located within the **Lane** ‘Prep’ of the **White-Box Pool** ‘New Car Sales’

The **SubProcess** ‘Close and Deliver’ is located within the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car Sales’

The **SubProcess** ‘Enter Order’ has an outgoing SequenceFlow to the **Exclusive Gateway** ‘Order from factory ?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’.

The **SubProcess** ‘Order Car from Factory’ has an outgoing SequenceFlow to the **Exclusive Gateway** ‘Car Unavailable ?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’.

The **Exclusive Gateway** ‘Order from factory ?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **SubProcess**’ Order Car from Factory’ when the **Gateway Condition Value** is ‘yes’.

The **Exclusive Gateway** ‘Car Unavailable?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **Terminate End Event** ’car unavailable’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ when the **Gateway Condition Value** is ‘yes’.

The **Exclusive Gateway** ‘Order from factory ?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **SubProcess**’ Prepare car for delivery’ when the **Gateway Condition Value** is ‘no’.

The **Exclusive Gateway** ‘Car unavailable ?’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **SubProcess**’ Prepare car for delivery’ when the **Gateway Condition Value** is ‘no’.

The **SubProcess** ‘Prepare Car for delivery’ has an outgoing SequenceFlow to a **Parallel Gateway** ‘1’ in the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car sales’.

The **Parallel Gateway** ‘1’ in the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **SubProcess** ’Close and deliver’.

The **Exclusive Gateway** ‘Financing unavailable ?’ in the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **Parallel Gateway** ‘1’ in the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car sales’ when the **Gateway Condition Value** is ‘no’.

The **Exclusive Gateway** ‘Financing unavailable ?’ in the **Lane** ‘Finance’ of the **White-Box Pool** ‘New Car sales’ has an outgoing SequenceFlow to the **Terminate End Event** ‘Financing unavailable’ in the **Lane** ‘Sales’ of the **White-Box Pool** ‘New Car sales’ when the **Gateway Condition Value** is ‘yes’.

The **SubProcess** ‘Enter Order’ has an outgoing SequenceFlow to the **SubProcess** ‘Arrange Financing’.

The **SubProcess** ‘Close and deliver’ has an outgoing SequenceFlow to the **Message End Event** ‘Success’ in the **Lane** ‘sales’ of the **White-Box Pool** ‘New Car sales’.

The **SubProcess** 'Arrange Financing' has an outgoing SequenceFlow to the **Exclusive gateway** 'Financing unavailable ?' in the **Lane** 'Finance' of the **White-Box Pool** 'New Car sales'

There is a Directional-Data Association for the **Data Object** 'Loan docs [approved]' starting from **SubProcess** 'Arrange Financing'

There is a Directional-Data Association for the **Data Object** 'Loan docs [executed]' starting from **SubProcess** 'Close and Deliver'.

There is a Directional-Data Association for the **Data Object** 'Loan docs [approved]' ending into **SubProcess** 'Close and Deliver'.

Figure 5.7 in [4] provides the expressions in a top-level process diagram. In order to be able to derive all relevant semantic associations between the concepts in the list of concept definitions for the level 1 palette we will add a second real-life example of an 'expansion diagram', as is for example given in figure 6-4 of Silver [4].

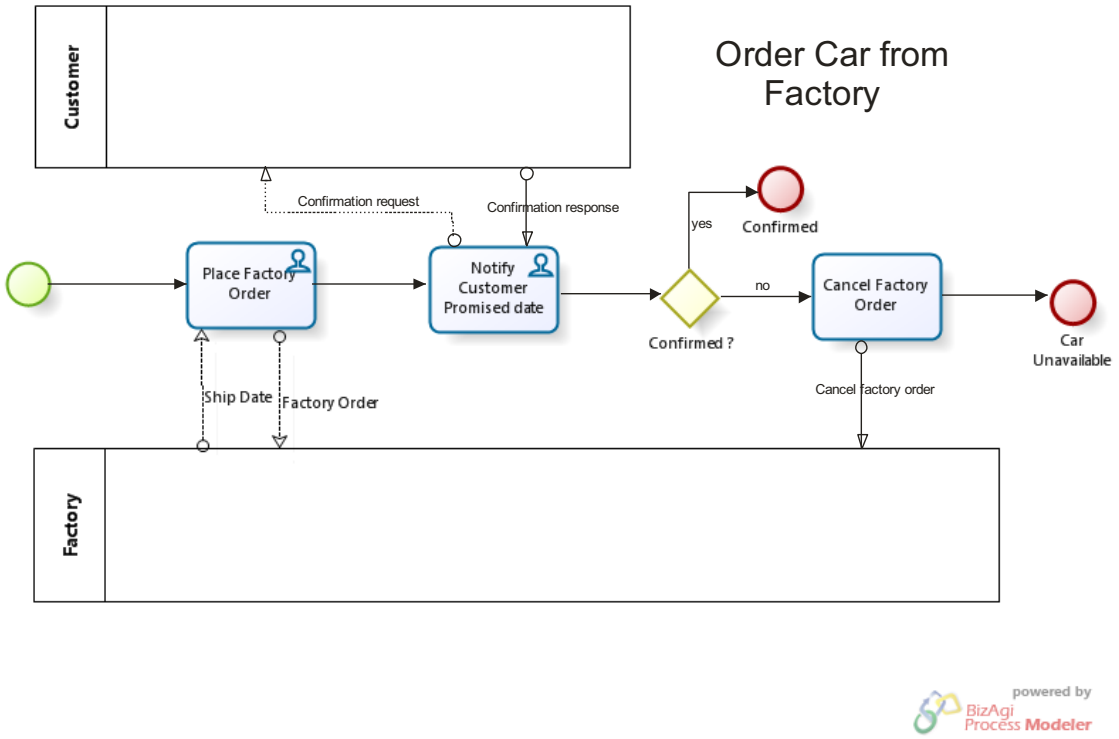

Fig 4. Example expanded BPMN model as in Silver [4]

The verbalization of the content of the example in figure 4 leads to the following ground fact sentences:

The **Task** 'Place factory order' within the **SubProcess** 'Order car from Factory' is a user task.

The **Task** 'Notify Customer Promised date' within the **SubProcess** 'Order car from Factory' is a user task.

The **Task** 'cancel factory order' within the **SubProcess** 'Order car from Factory' is a user task.

A **None Start Event** has an outgoing Sequence Flow to the **Task** 'Place factory Order' that is defined within the **SubProcess** 'Order car from Factory'.

The **Task** 'Place Factory order' within the **SubProcess** 'Order car from Factory' has an outgoing **MessageFlow** 'Factory order' to the **Black-Box Pool** 'Factory'.

The **Black-Box Pool** 'Factory' has an outgoing **MessageFlow** 'Ship date' to the **Task** 'Place Factory order' that is defined within the **SubProcess** 'Order car from Factory'.

The **Task** 'Notify Customer Promised Date' within the **SubProcess** 'Order car from Factory' has an outgoing **MessageFlow** 'Confirmation Request' to the **Black-Box Pool** 'Customer'.

The **Black-Box Pool** 'Customer' has an outgoing **MessageFlow** 'Confirmation response' to the **Task** 'Notify Customer promised date' that is defined within the **SubProcess** 'Order car from Factory'.

The **Task** 'Place Factory Order' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **Task** 'Notify Customer Promised date' within the **SubProcess** 'Order car from Factory'.

The **Task** 'Cancel Factory order' within the **SubProcess** 'Order car from Factory' has an outgoing **MessageFlow** 'cancel factory order' to the **Black-Box Pool** 'Factory'.

The **Exclusive Gateway** 'Confirmed ?' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **Task** 'cancel factory order' within the **SubProcess** 'Order car from Factory' when the **Gateway Condition Value** is 'no'.

The **Exclusive Gateway** 'Confirmed ?' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **None End Event** 'confirmed' within the **SubProcess** 'Order car from Factory' when the **Gateway Condition Value** is 'yes'.

The **Task** 'Notify Customer Promised Date' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **Exclusive gateway** 'confirmed?' within the **SubProcess** 'Order car from Factory'.

The **Exclusive Gateway** 'Confirmed ?' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **None End Event** 'confirmed' within the **SubProcess** 'Order car from Factory' when the **Gateway Condition Value** is 'yes'

The **Task** 'Cancel factory order' within the **SubProcess** 'Order car from Factory' has an outgoing SequenceFlow to the **Terminate Event** 'car unavailable' in the **Lane** 'sales' of the **White-Box Pool** 'New Car sales'.

We have now fully verbalized this significant example. The resulting collection of sentences will allow us to reconstruct the original example using the telephone metaphor [18].

3.4 Abstracting the ground facts into an Information Structure Diagram (ISD)

In this section we will show the results of grouping the ground facts and abstracting them into 'fixed parts' (or verbs) and the 'variable parts' (or roles). The results of this transformation is a 'role-box' diagram that contains the abstraction of the ground facts and an example population in which the 'variable names' in

the ground facts are listed (see figures 4 and 5). Note that we have given the fact types that are derived directly from the verbalizations in section 2, a fact type number in the range from 1 to 100. Fact types that encode specializations of a 'super'- concept into 'sub'-concepts have been assigned a fact type number in the 200's. The fact types that encode a compound naming convention (see section 3.2) for a specific concept have been give fact type numbers in the range from 101 to 199.

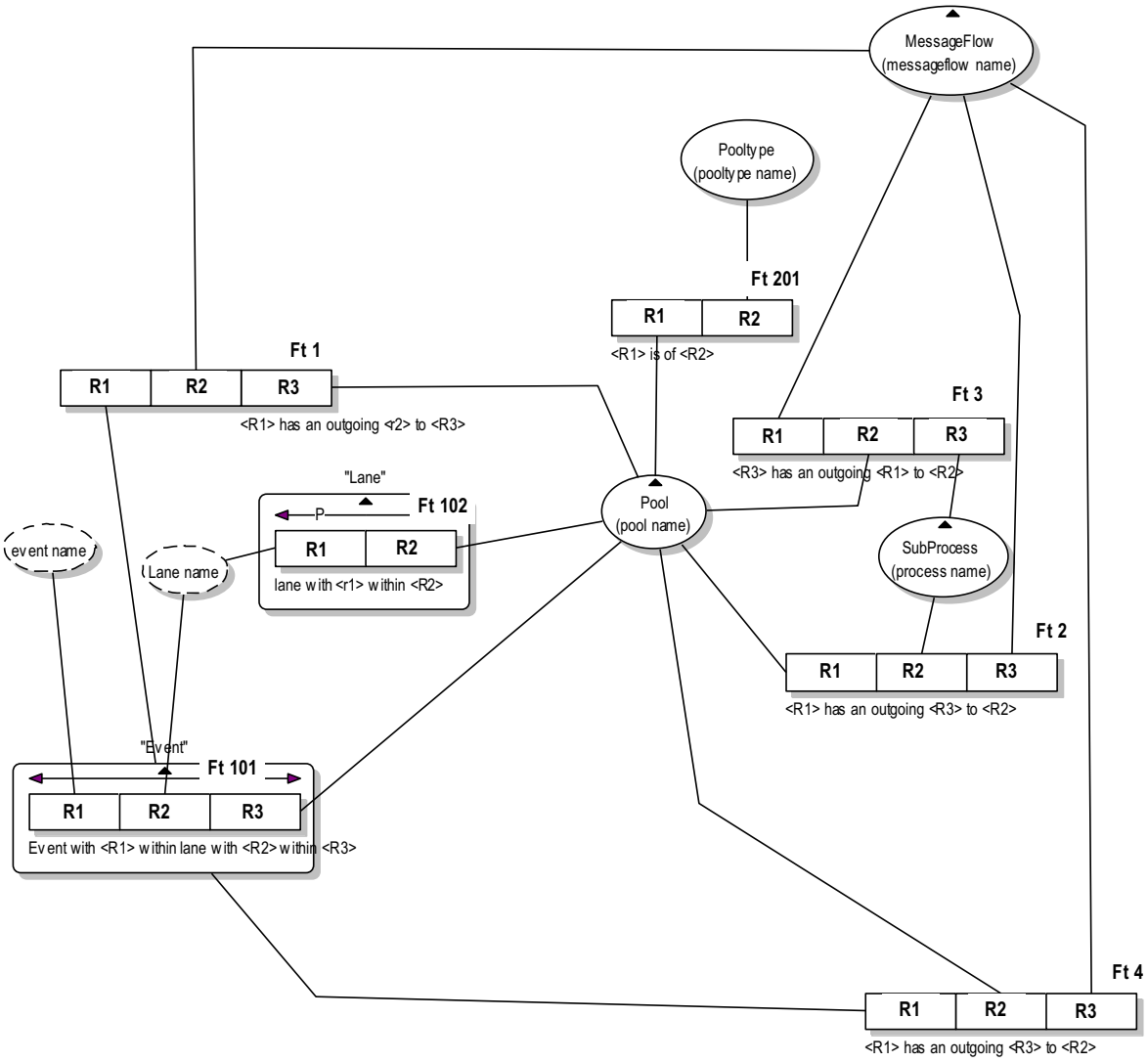

Fig 5a. Information Structure diagram for BPMN palette 1 modeling constructs (I).

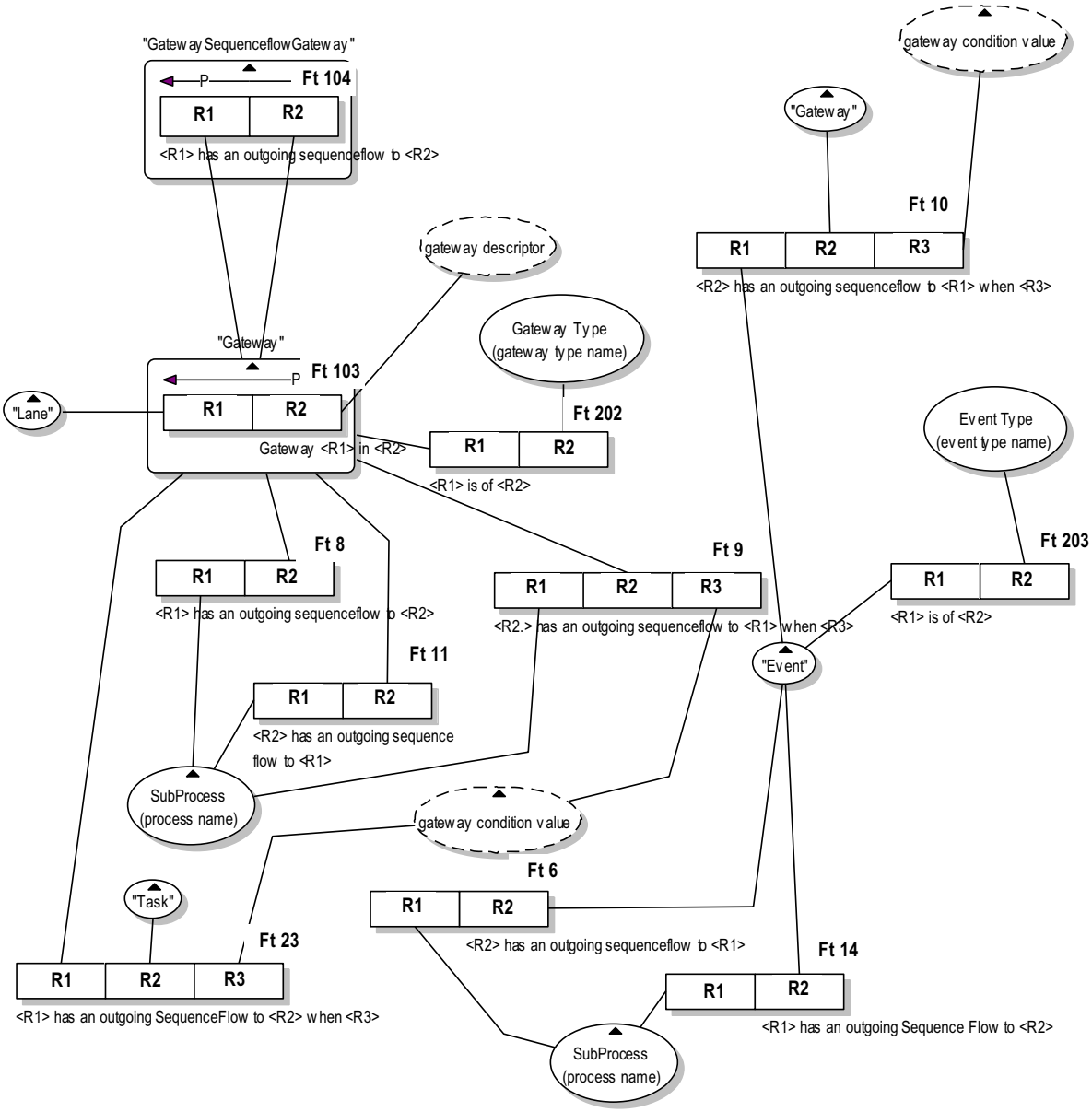

Fig 5b. Information Structure diagram for BPMN palette 1 modeling constructs (II).

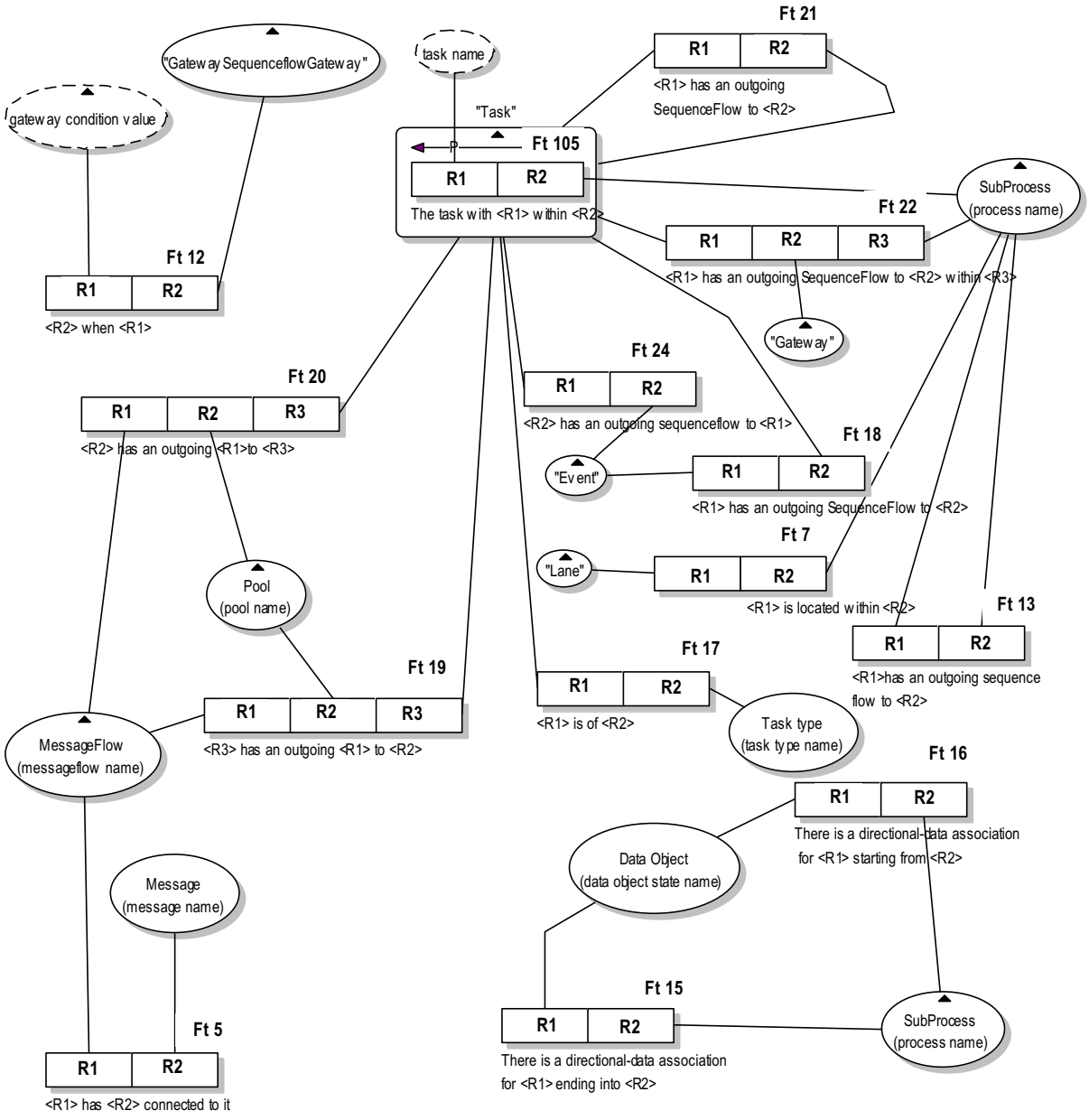

Fig 5c. Information Structure diagram for BPMN palette 1 modeling constructs (III).

In figures 5a, 5b and 5c we have given the information structure diagram with the abstracted domain fact types, sub-type fact types and the 'compound' naming convention fact types. Because an information structure diagram can be considered to be a communication document within the domain of a business analyst we have introduced the naming convention for roles and fact types. In this way a domain ISD can be verbalized and analyzed, for example to see whether it complies to the meta model. In figure 6 we have given an excerpt of our example ISD for the 'BPMN happy path' including the population of the fact types.

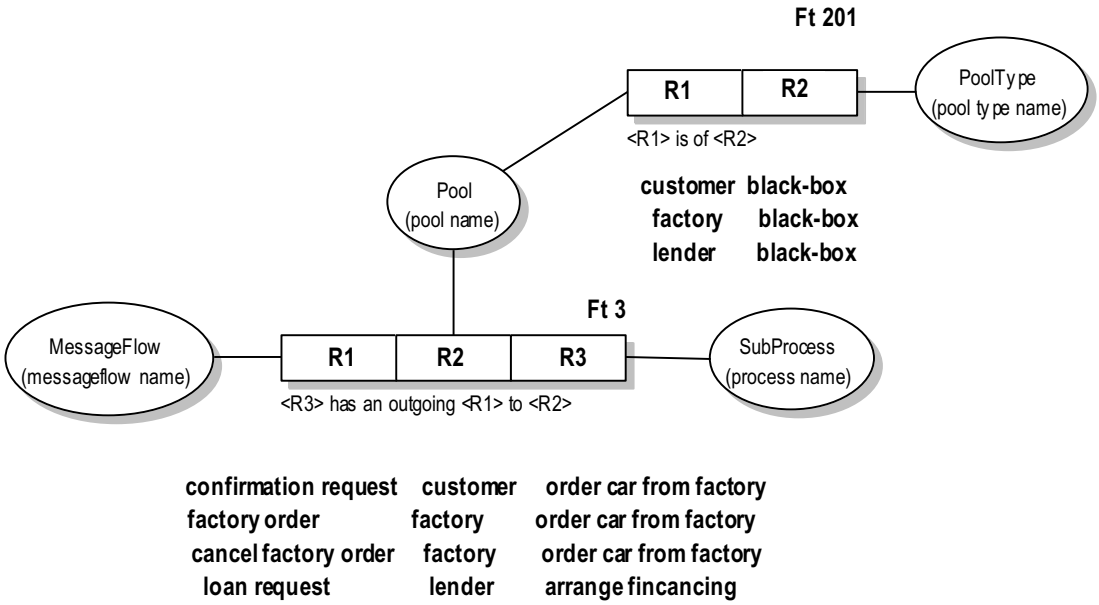

Fig 6. Populated Information Structure Diagram for BPMN example excerpt

The content of the excerpt from figure 6 is equivalent to the following set of ground fact sentences from section 3.3:

The **SubProcess** 'order Car from Factory' has an outgoing **MessageFlow** 'confirmation request' to the **Black-Box Pool** 'Customer'

The **SubProcess** 'Order Car from factory' has an outgoing **MessageFlow** 'Factory order' to the **Black-Box Pool** 'Factory'.

The **SubProcess** 'order Car from Factory' has an outgoing **MessageFlow** 'Cancel Factory order' to the **Black-Box Pool** 'Factory'.

The **SubProcess** 'Arrange Financing' has an outgoing **MessageFlow** 'Loan request' to the **Black-Box Pool** 'Lender'.

3.5 Deriving additional Business Rules from an Information Structure Diagram (ISD)

The next step in the fact-oriented modeling methodology consists of deriving and adding business rules to the information structure diagram. In this article we will, apply the explicit derivation steps from the conceptual schema modeling procedure [5, 18]. We will call this way of deriving business rules the 'inside-out' approach. Another way of finding the instances of these constraints is to interpret textual/diagrammatic descriptions (the outside-in approach).

3.5.1 The derivation of constraints using the CSDP (the inside-out approach)

We will start to analyze the fact types in our ISD that originate in the verbalizations of the BPMN example, and therefore are the fact types which numbers lie in the range Ft1 through Ft 100. Fact type Ft1 and it's (initial) example population is given in figure 7.

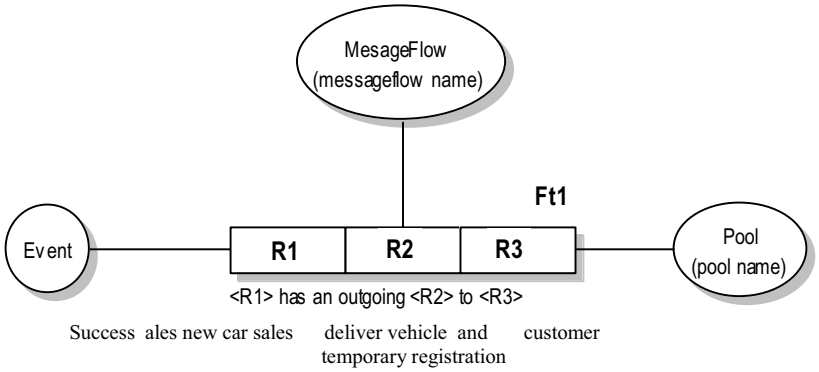

Fig 7. Populated Information Structure Diagram for fact type FT1

In our BPMN example, only one instance of fact type Ft1 is encountered. This means that we will have to create a significante example by adding different instances of outgoing message flows in a newly constructed model.The methodology works as follows. Take an allowed sentence.

The **Message End Event** 'Success' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing **MessageFlow** 'Deliver vehicle and temporary registration' to the **Black-Box Pool** 'Customer'……(sentence 1)

Let's adapt this sentence into a second sentence as follows by changing the value of role R1:

The **Message End Event** 'New buyer' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing **MessageFlow** 'Deliver vehicle and temporary registration' to the **Black-Box Pool** 'Customer'……(sentence 2)

Adding the first and second sentence into a 'constructed excerpt' of a BPMN model will lead to the process model example in figure 7.

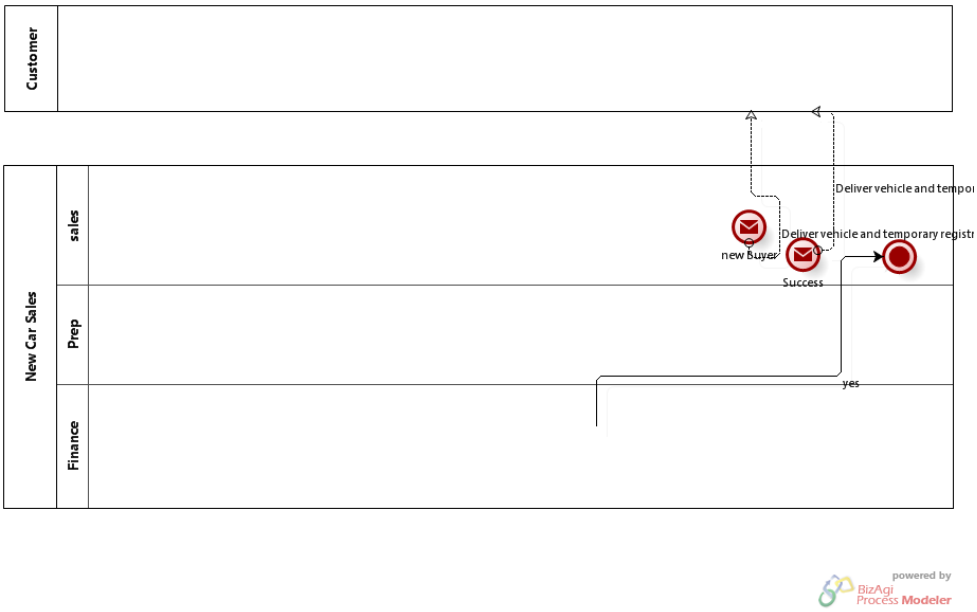

Fig 8. Constructed BPMN model for checking the co-existence of sentences (1) and (2)

Inspecting this example in figure 8, tells us that it is an allowed instance of a BPMN process model. We will therefore, add the sentence to the significant set of sentence instances and we will create a third example sentence. This third example sentence can be created by taking the first sentence once again, but now changing the value of role *R2*.:

The **Message End Event** 'Success' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing **MessageFlow** 'deliver brochure' to the **Black-Box Pool** 'Customer'……..(sentence 3)

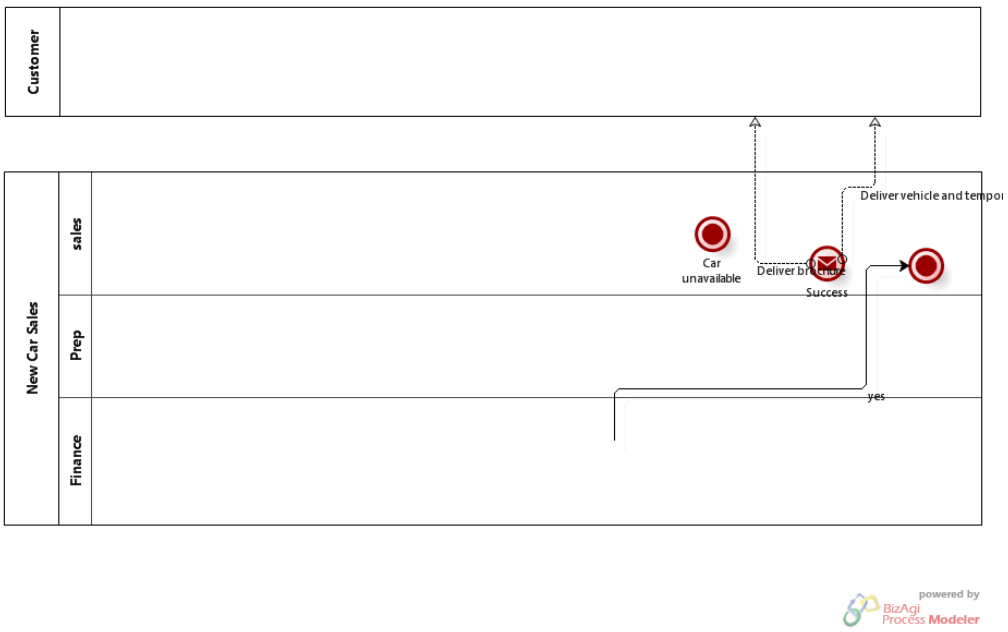

Fig 9. Constructed BPMN model for checking the co-existence of sentences (1) and (3)

The BPMN analyst considers the example in figure 9 as an example that is **not** allowed to occur, because the message semantics between an end event and a white-box pool should be a pooled one (as is given in figure 10).

Fig 10. Allowed BPMN model for the semantics of sentences (1) and (3)

The fact that the third sentence is not allowed to co-exist with sentence 1, implies that we have found a uniqueness constraint *C1* that spans *all* roles except role *R2*. The last step in the analysis of uniqueness constraints for fact type Ft1 is the checking of the third adapted sentence with the first one (sentence 4). We will create this sentence by changing the value of role R3 and check the coexistence of this sentence with sentence 1:

The **Message End Event** 'Success' within the **Lane** 'Sales' of the **White-Box Pool** 'New Car Sales' has an outgoing **MessageFlow** 'Deliver vehicle and temporary registration' to the **Black-Box Pool** 'Lender'……..…(sentence 4)

In the defining document of the BPMN standards, we can not find any rule that would forbid a creation of a BPMN process model as given in figure 11. This means that sentence 4 can coexist with sentences 1, 2 and 3.

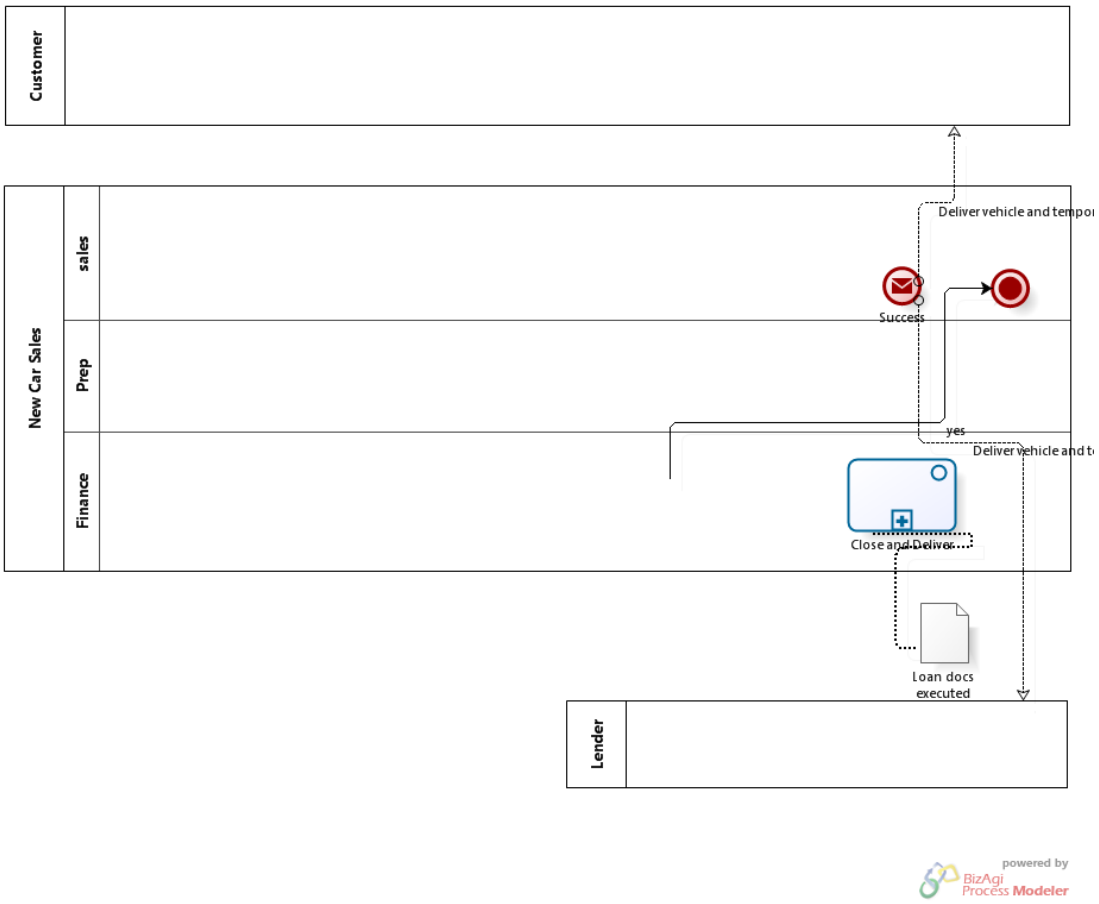

Fig 11. Allowed BPMN model for the semantics of sentences (1) and (4)

We now have finished analyzing the possible configurations of intra-fact type uniqueness constraints for fact type Ft1. We conclude that we have discovered 1 uniqueness constraint (*c17*) for fact type *Ft1* that spans roles *R1* and *R3* (see figure 12).

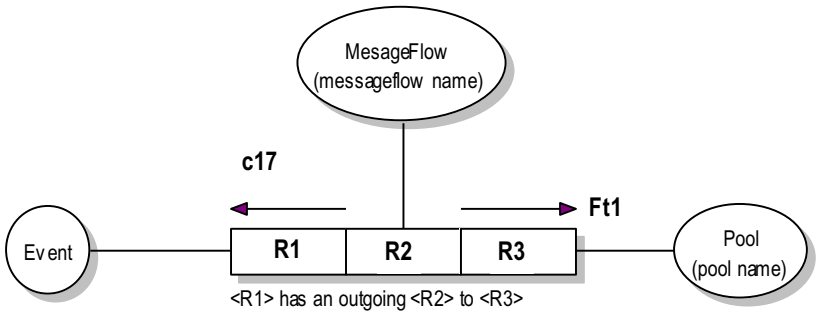

Fig 12. Information Structure Diagram for fact type FT1 including Uniqueness Constraint

If we inspect an allowed example in figure 4-10 of [4], we see that it is allowed that two different sequence-flows coming out from two different subprocess can enter the same gateway. This implies that the potential uniqueness constraint on role R2 of fact type *Ft8* does NOT exist. After carefully inspecting many examples of BPMN models in the OMG standard and Silver's book we did not find a situation in which there's are two different outgoing sequence flows each to a different gateway. This makes sense because a token or process instance can only follow one path at the same time. However, this modeling constraint is very hard to 'trace' in the standard documentation and the text book we examined. So in figure 13 we have given an excerpt of BPMN model that is **not** allowed to exist.

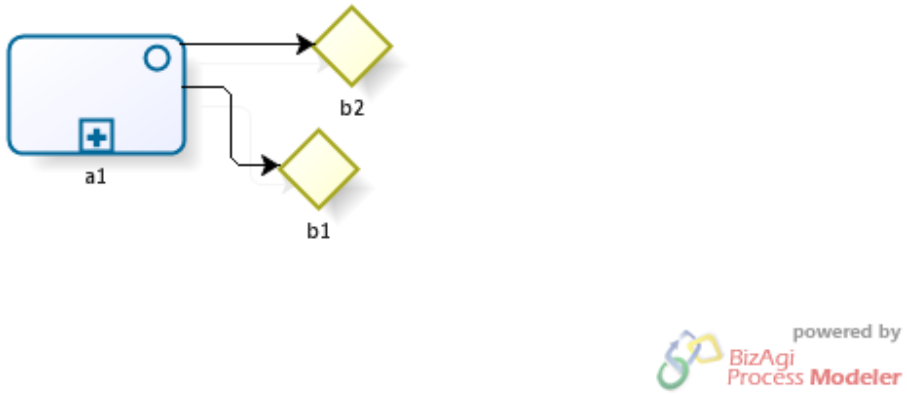

Fig 13. Constructed Non-allowed BPMN model for checking the co-existence of sentences (11) and (12)

Analogue to the derivation of the excerpt of the fact types in BPMN meta model we can create an example diagram based upon the following verbalization:

The **SubProcess** 'a1' has an outgoing SequenceFlow to **Gateway** 'b1'……(sentence 11)
The **SubProcess** 'a1' has an outgoing SequenceFlow to **Gateway** 'b2'……(sentence 12)

Sentences 11 and 12 cannot exist at the same time, to depict this we have defined uniqueness constraint *C3* on role *R1* of fact type *Ft8* in the BPMN meta model (see figure 14).

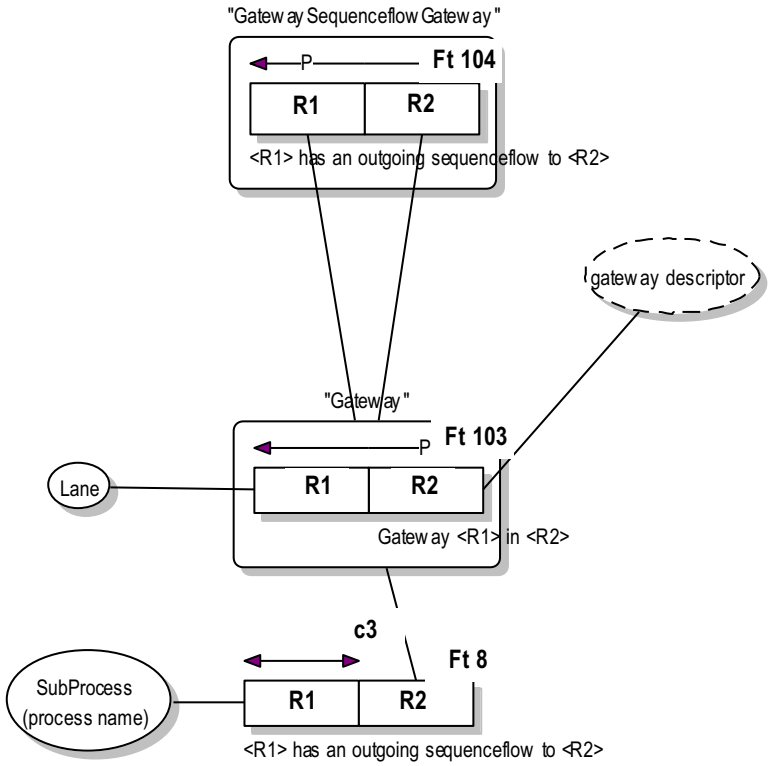

Fig 14. Information Structure Diagram for fact type Ft8 including Uniquness constraint constraint *c3*

With respect to fact type *Ft11*, we find in the example BPMN in figure 3, that all sentence permutations are allowed to exist e.g., the following sentences (13,14 and 15) are allowed to exist in combination:

Gateway 'b1' has an outgoing SequenceFlow to **Subprocess** 'a1'……..……(sentence 13)
Gateway 'b2' has an outgoing SequenceFlow to **Subprocess** 'a1'……..……(sentence 14)
Gateway 'b1' has an outgoing SequenceFlow to **Subprocess** 'a2'……..……(sentence 15)

The diagrammatic equivalent expression of these sentences can be found in figure 15.

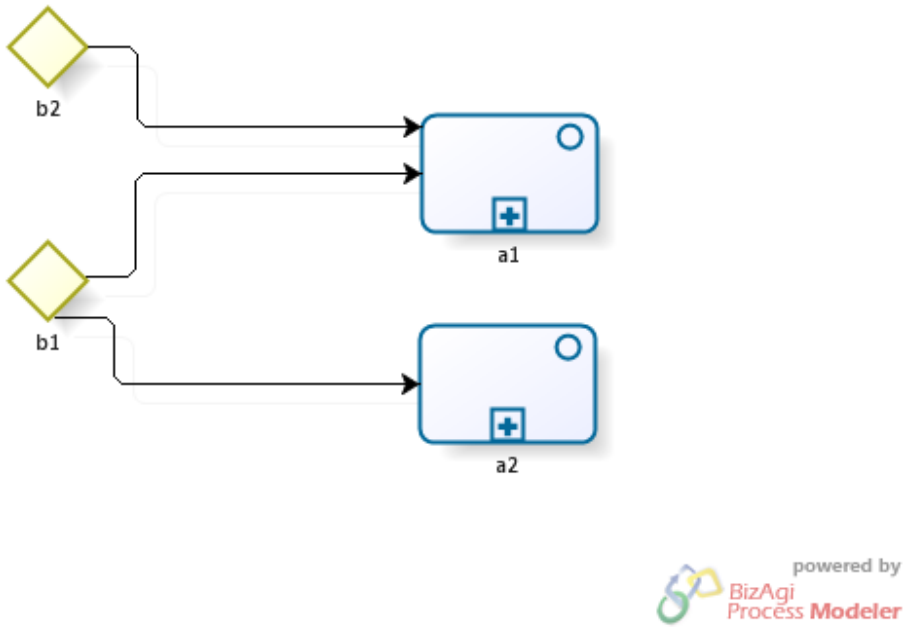

Fig 15. Constructed allowed BPMN model for checking the co-existence of sentences (13), (14) and (15)

So the absence of the potential uniqueness constraint defined on role *r1* of fact type *Ft11* in the BPMN meta-model excerpt in figure 5b in combination with the absence of the potential uniqueness constraint defined on role *r2* of fact type *Ft11*, implies that we have found uniqueness constraint *c4*, spanning both roles of meta fact type *Ft11*.

If we inspect the text book of Silver [4], we find an example in figure 9.7 in which the following sentences can derived:

SubProcess 'pre process phone order' has an outgoing SequenceFlow to **SubProcess** 'fulfill order'……..(sentence 16)
SubProcess 'pre process web order' has an outgoing SequenceFlow to **SubProcess** 'fulfill order'……..(sentence 17)

On the other hand we did not find examples of the following sentence combinations:

SubProcess 'pre process phone order' has an outgoing SequenceFlow to **SubProcess** 'fulfill order'……..(sentence 17)
SubProcess 'pre process phone order' has an outgoing SequenceFlow to **SubProcess** 'reject order'……..(sentence 18)

The reason for this is that one process instance can only follow one path. This leads to the creation of uniqueness constraint *c5* of fact type Ft15.

3.5.2 The addition of constraints using the outside-in approach

A second way of 'discovering' uniqueness constraints is to take textual or diagrammatic documents, e.g. the BPMN standard document [2] or an explanatory text book, e.g. Silver [4] and 'translate' a textual segment as an uniqueness constraint on a fact type from the ISD. For example on page 40 of [2] the following text can be found:

'A start event MAY be the target for Message Flow; it can have 0 (zero) or more incoming Message Flow.'

This statement basically tells us that there is for sure *no uniqueness* constraint defined on role *R2* of fact type Ft14 and that there is for sure no uniqueness constraint defined on roles *R1 and R3* of fact type Ft10.

On page 73 of the BPMN standard document [2] we find the following description:

'Each gate must have an associated outgoing sequence flow'. This can be modelded as a mandatory role constraint *C2* for the object type gateway in role *r2* of fact type *Ft9* (see figure 16).

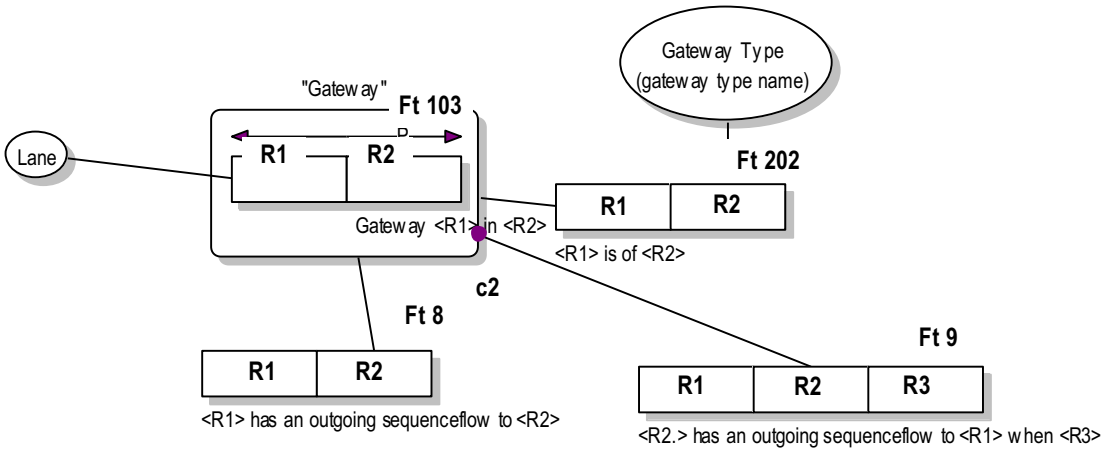

Fig 16. Information Structure Diagram for fact type FT8, Ft9, Ft102 and Ft103 including mandatory role constraint *c2*

3.6 The BPMN meta model including population constraints.

In this paper we have illustrated for a few of these constraints how they could be derived by applying the fact-oriented knowledge extraction procedure or CSDP. In figure 17 we have shown the completed fact-oriented meta model for the palette 1 modeling constructs in BPMN.

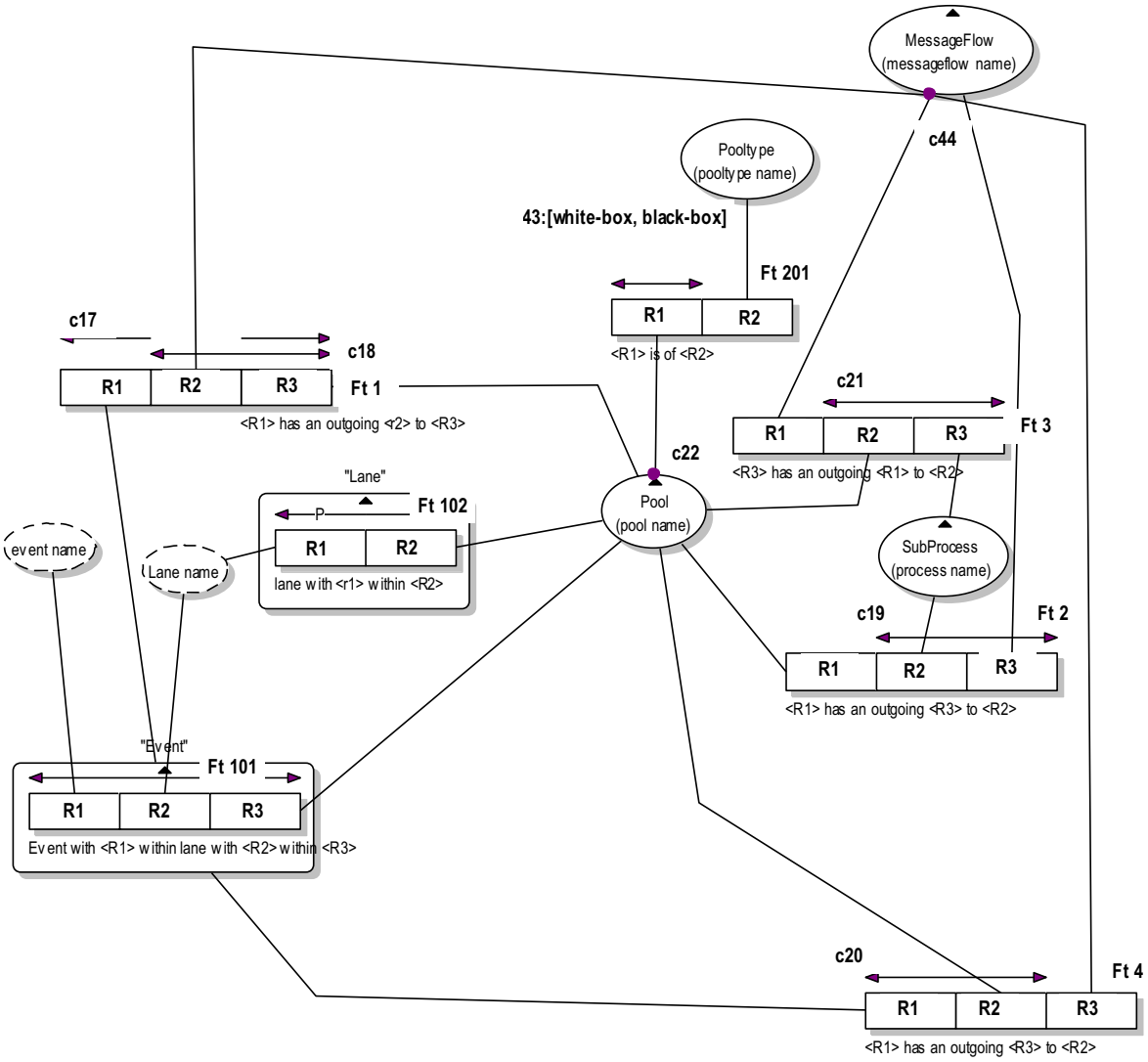

Fig 17a. Information Structure diagram for BPMN palette 1 modeling constructs including constraints (I).

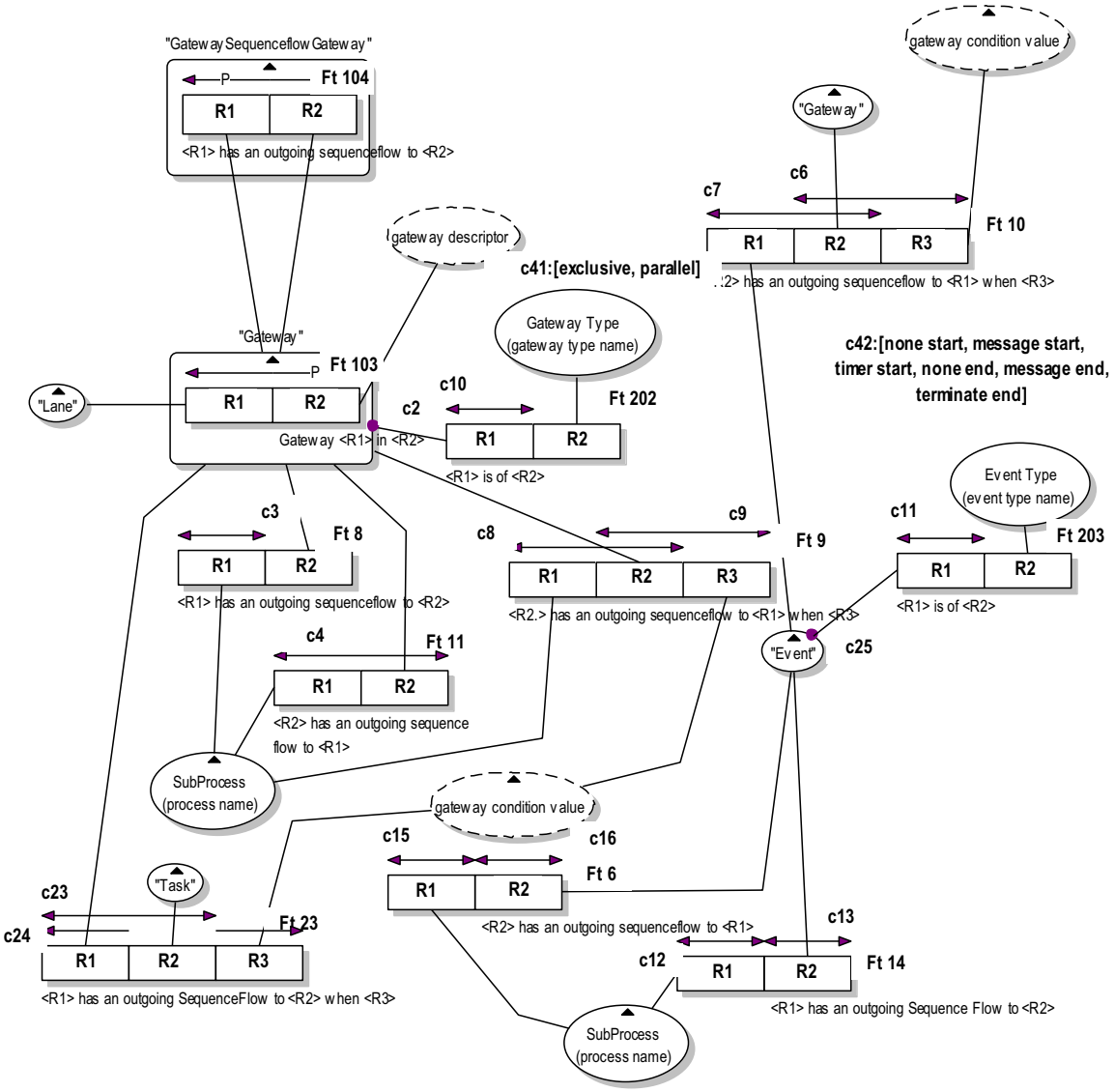

Fig 17b. Information Structure diagram for BPMN palette 1 modeling constructs including constraints (II).

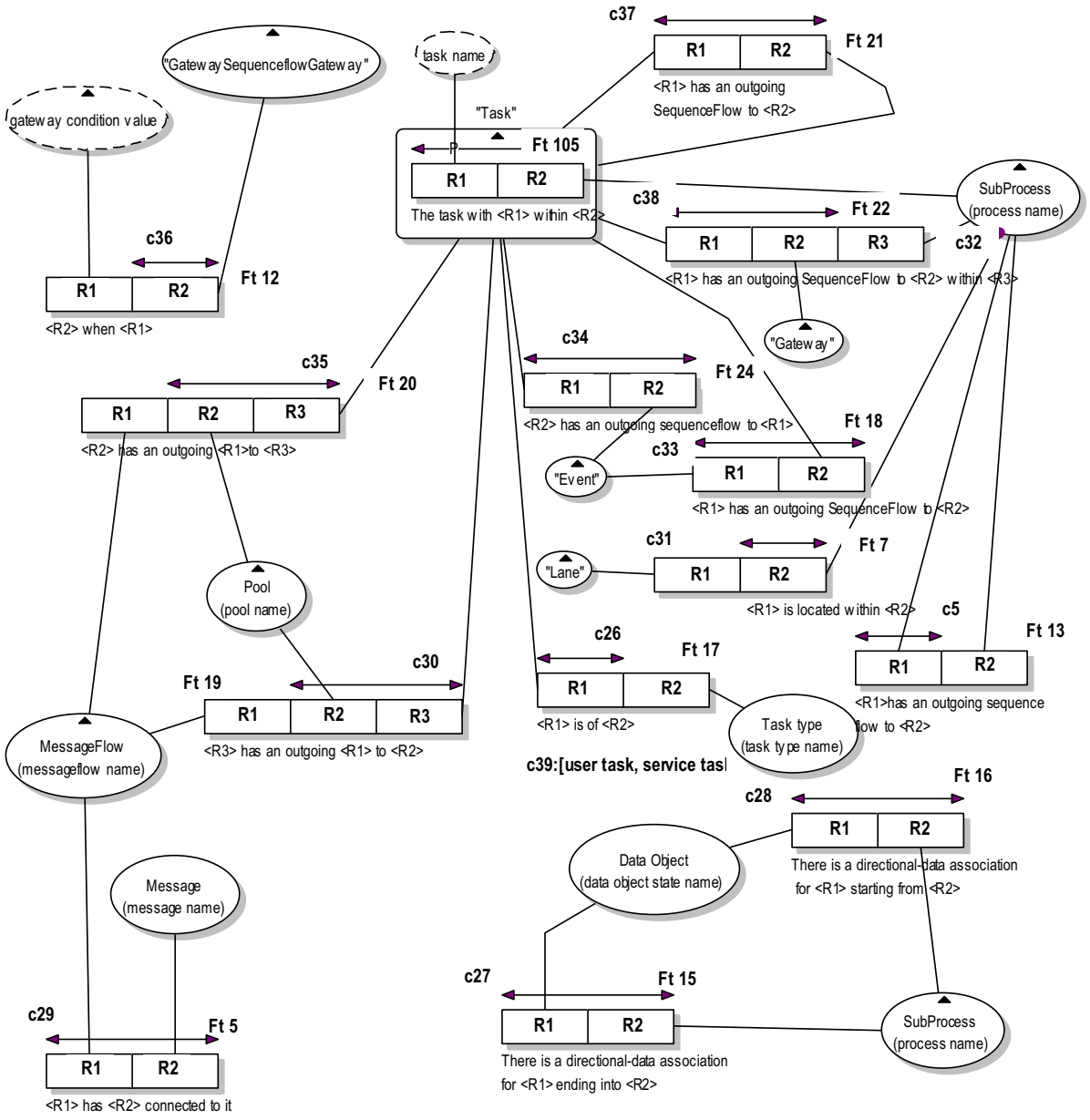

Fig 17c. Information Structure diagram for BPMN palette 1 modeling constructs including constraints (III).

4. Conclusions

In this paper we have illustrated how a fact-oriented modeling methodology can help business analyst, standard developers, and tool-designers in capturing essential 'business rules' for applying (in this case) the BPMN process modeling standard. Although the standard document and a recent practitioner's text book were studied in detail, we have shown that essential modeling constraints were derived from (the absence of) allowed BPMN model examples. Although we have limited ourselves in this paper to the palette 1 modeling constructs for BPMN, the approach is fully scalable to cover the 'complete' BPMN

standard or any other knowledge domain, as long as we restrict ourselves to 'verbalizable information'. The advantage of using the fact-oriented modeling methodology lies amongst other things in the strengths of using examples. As we have illustrated in this article, it is the 'semantic richness' of tangible examples or 'data' use cases combined with the information structure diagram (ISD). On a couple of occasions we have illustrated how the ISD in combination with allowed diagrammatic expressions 'expose' the domain rules that govern the instances of the BPMN modeling standard in terms of uniqueness and mandatory role constraints that are not explicitly documented in the definingBPMN literature.

Current tools that support fact-oriented modeling are DocTool [19] and NORMA [20]. These tools provide modeling support for large models and basically are fully scalable in terms of size and complexity.

References

1. OMG, *Semantics of Business Vocabulary and Business Rules (SBVR), v1.0 OMG Available Specification*. 2008.
2. OMG, *Business process modelling notation (BPMN) OMG available specification v 1.1.* 2008, OMG.
3. White, S.A. (2006) *Introduction to BPMN*. On demand business
4. Silver, B., *BPMN Method & Style*. 2009, Aptos, California, USA: Cody-Cassidy Press.
5. Halpin, T. and T. Morgan, *Information Modeling and Relational Databases; from conceptual analysis to logical design* 2nd ed. 2008, San-Francisco, California: Morgan-Kaufman.
6. Nijssen, G. and T. Halpin, *Conceptual schema and relational database design: a fact oriented approach*. 1989, New-York: Prentice-hall. 337.
7. Bollen, P. *The Natural Language Modeling Procedure*. in *5th international workshop on Next Generation Information Technologies and Systems*. 2002. Caesarea, Israel: Springer.
8. Nijssen, G. and R. Bijlsma. *A conceptual structure of knowledge as a basis for instructional designs.* . in *The 6th IEEE international conference on Advanced Learning Technologies, ICALT 2006*. 2006. Kerkrade, the Netherlands.
9. Halpin, T., *Information Modeling and Relational Databases; from conceptual analysis to logical design*. 2001, San Francisco, California: Morgan Kaufmann.
10. Nijssen, G. *On the gross architecture for the next generation database management systems*. in *Information processing 1977*. 1977: IFIP.
11. Verheijen, G. and J. van Bekkum. *NIAM: An Information Analysis method*. in *IFIP TC-8 CRIS-I conference*. 1982: North-Holland, Amsterdam.
12. Bakema, G.P., J.P. Zwart, and H. van der Lek, *Fully communication oriented NIAM*, in *NIAM-ISDM 1994 Conference*, G. Nijssen and J. Sharp, Editors. 1994: Albuquerque NM. p. L1-35.
13. Nijssen, G., *Kenniskunde 1A*. Vol. PNA Publishing Heerlen. 2001.
14. Lemmens, I., M. Nijssen, and G. Nijssen, *A NIAM 2007 conceptual analysis of the ISO and OMG MOF four layer metadata architectures*, in *OTM 2007/ ORM 2007*. 2007, Springer: Vilamoura, Algarve, Portugal.
15. Nijssen, G. and J. Hall, *SBVR diagram; a response to an invitation.* Business Rules Journal, 2008. **9**(7).
16. Bollen, P., *SBVR: a fact oriented OMG standard*, in *OTM workshops 2008* 2008, Springer Verlag: Monterry mexico.
17. Halpin, T. and M. Orlowska, *Fact-oriented Modeling for Data Analysis.* Journal of Information Systems, 1992. **2**: p. 97-118.
18. Nijssen, G., *Grondslagen van Bestuurlijke Informatiesystemen*. 1989, Slenaken: Nijssen Adviesbureau voor Informatica.
19. Nijssen, M., I. Lemmens, and R. Mak, *Fact-Orientation Applied to Develop a Flexible Emplyment Benefits System*, in *ORM 2009*. 2009, Springer: Vilamoura, Portugal.
20. Halpin, T., *Predicate Reference and Navigation in ORM*, in *ORM 2009*. 2009, Springer: Vilamoura, Portugal.